HST Control Center System

Engineering Data Archive Study

The Problem

Current projections for the volume of data to be stored by the CCS archive are on the order of 1.5 Terabytes per year, expanding to upwards of 30 Terabytes for an extended 20-year mission. This estimate varies depending on factors such as telemetry format, archive format, and volume of ancillary data to be stored. Storage, management, and reasonable use of this amount of data poses a technological challenge.

Access to at least a subset of this data is presently based on a data warehousing product. The choice of this technology places certain constraints on the storage and management solutions. Additionally, system resources present constraints, or at least require consideration prior to development.

It is the intent of this study to provide a recommended approach to formatting and storing the archive data, such that it is compatible with the data warehouse product, provides reasonable performance, does not represent an unreasonable impact to system resources, and does not overly complicate system development, operations, and maintenance.

The System

The system is comprised of four elements: Preloader, Data Warehouse, Data Server, and Ancillary Storage.

Preloader

The preloader receives data from Merge Process, which is assumed to be in FEP Output Format (FOF) packets. The preloader's task is to convert data from the packetized FOF to fields for input to the data warehouse. For all options this requires at least associating each mnemonic and its values with time and data quality information.

Dependent upon constraints of the data warehouse, only a subset of the entire data stream can be stored internally. The preloader, therefore, must also extract the subset to be stored, and forward any additional data to ancillary storage. Additional pre-processing may be required, depending upon the storage format.

Data Warehouse

The data warehouse receives data from preloader, ingests it into tables, maintains associations, manages the warehouse storage resource, processes queries from data server, and outputs solution sets to data server in DW output format. Processing and storage volume depend upon storage format.

Data Server

The data server receives and manages query requests from users, reformats if necessary, and submits queries to the DW. Solution sets are received from the DW and formatted into CDF packets for transmission to user. Output format details (i.e. changes only/all points, integer/float, raw/EU) are dependent upon definition of CDF and design of the data server. Nature of processing required to format the output is dependent upon design decisions yet to be made, and storage format to be chosen.

Ancillary Storage

Ancillary storage stores and maintains any data which is not kept in the data warehouse. This includes telemetry data which is excluded from the data warehouse by virtue of the choice of format, user files, macros, init files, solution sets required to be kept, and other data which may be dependent on the storage format.

The Data

The telemetry data exists in four basic formats: FEP Output Format (FOF), Common Data Format (CDF), Data Warehouse Format, and Ancillary Storage Format.

FEP Output Format

The FOF is the format of data transmitted by the FEP, and received by the Merge Processor. FOF files are temporarily staged under control of the Merge Processor. At some point in time, all data in the staging area is merged into a "seamless" set of FEP packets and forwarded to the archive's Preloader process.

Each FOF packet received by the Preloader consists of a header object and an element object. The header contains three time fields, a telemetry format, and data source information. The element object contains a variable number of records consisting of mnemonic identifier, raw PCM counts, EU converted value, and data quality flags. Two packets will exist for every minor frame of telemetry (high rate format) which is received by the FEP.

Common Data Format

The CDF is the format of data transmitted by the data server to the user. The construction of a set of CDFs is performed in response to a user request for data. The solution set which satisfies the query will be composed into CDF format by a data server process, and then transmitted to the user process.

Each CDF packet consists of a header object and an element object. The header contains a single time field, a telemetry field, and data source information. The element object contains a variable number of records, each consisting of mnemonic identifier, data type indicator, data field, and flags. The number of CDF packets will depend upon the nature of the query and the time span requested.

The data can be transmitted to the user as all contiguous data points, as changes only, or the user can select either.

Ancillary Storage Format

Ancillary data consists of engineering telemetry which is not stored in the data warehouse, but is used to support data queries and retrievals. This data may include, but is not limited to, high-rate and OBC memory parameters, dump data, derived data, etc. This data is generally kept in a flat file format.

Ancillary storage may also include additional, non-telemetry data which is to be centrally managed by the archive system.

The Issues

The issues under consideration concern the cost of hardware and media, versus resource utilization and performance, versus the complexity of software required to implement and support data reduction and reconstruction. Several storage format options were evaluated during this study; those which were eliminated are discussed in Appendix A. Three specific issues were considered in reaching a final recommendation. Those issues were:

1. Data Warehouse Storage Format - what storage format will provide a reasonable balance among the cost, performance, and complexity criteria.
2. Ancillary Data Storage - all warehouse storage options still place a high demand on ancillary data storage; what options exist to maintain this data. What are the choices and costs of minimizing this data.
3. Archive Architecture - the choice of architecture is a cost issue, but also has implications on long term maintenance and media storage. What architecture supports an open system at reasonable cost and performance. What storage management options exist.

Data Warehouse Storage Format

Two alternative storage methods have been considered: store only the data which changes (changes-only), or store all points of data directly (all-points). Changes-only maintains a value and the period of time represented by that value, whereas all-points stores all values at a given point in time. These two methods are mutually exclusive, since they represent two orthogonal views of the same information: time-domain vs. frequency-domain.

These two options have opposing advantages and disadvantages, making neither one a clear choice. Each one represents an advantage to its own class of users, who have a secular view of the data. Performance of the system can be tuned to one type of query only at a cost to the other. Several variations of these options were considered in this analysis, and it became apparent to the community that the final format needs to be some combination of the two options.

Regardless of the storage method proposed, the volume of data must be reduced to a subset which is manageable by the warehouse. This limitation requires that the most dynamic data be stored outside the data warehouse, and some useful representation be stored in the warehouse for queries. Therefore, the recommended solution is to split the telemetry stream based upon nominal rates of change. Parameters which typically change slowly will be stored in the data warehouse. More dynamic parameters will be stored in flat files in ancillary storage, with their averages (or some alternative; e.g. culled) stored in the warehouse.

This scheme raises an issue for further consideration, however. The choice of parameters to be stored in the warehouse is somewhat arbitrary; i.e. it is based upon data characteristics or perceived usefulness, criteria which may change as the system matures. This implies that some configuration control mechanism will be implemented to manage the data warehouse content. This issue has not been addressed.

Issue #1 - Configuration management of the content of the database must be resolved. How are parameters selected for inclusion or exclusion. What criteria are applied to the choice. How will exceptions be resolved and implemented.

A second issue concerns the design of the database itself. The Red Brick product currently limits the amount of data which is directly accessible to about 1.5 billion records. This limitation has not been tested operationally, however, so an effort is underway to determine the performance ceiling of this product. The results of this benchmark may help better define the subset of data which will be selected to be managed by the data warehouse.

Red Brick allows tables to be segmented, along with their indices. This allows data to roll off periodically in order to keep the size of the database manageable. Queries which hit off-line data segments can return partial solution sets without that data, or the tool can suspend the query while off-line segments are reloaded. This segmentation works well for simple time indexed data, but becomes complex for other schema.

Performance rapidly becomes an issue as the database design becomes more complex. Conversely, if the data is strictly limited to a very few parameters, the utility of the database is questionable. This issue also remains to be resolved.

Issue #2 - The database design is currently in development. Preliminary results have shown limitations. Choice of data format and selection of parameters is dependent upon database design, and the design and capabilities of the data warehouse may change at any time. This represents a risk to the design of the data server. Designs of the data server and data warehouse must be encapsulated in order to avoid impacting other elements of the archive subsystem.

Table 1 indicates estimates for growth of data volume, and number of database records. This data predicts, first, that changes-only data with high-rate averaged (or culled) data accounts for 84 Gbytes per year. This volume is projected through 2011, assuming current data rates. The first five years of data is estimated to be 4/5 low-data-rate (8 kbps).

Analysis also shows that a 1.5 billion-row table will support 322 days of changes-only data, and would require 72 GBytes of storage. This rate translates to 141 million rows per month (6 Gbytes), which may be a reasonable segment size.

The volume projection assumes a 32-byte record length (reflecting two time fields and an 8-byte EU field), and a 50% index overhead. Pilot database designs have ranged from 30% to 150% index overhead.

Assumptions are contained in column 1. Not all assumed values are used in the portion of the spreadsheet represented by Table 1.

Table 1 - Data Warehouse Estimates, Changes Only

6472	params per maj frame
161.445	avg data points / mframe [1]
20	mframes / sec
3228.9	data points / sec (pps)
32	bytes / data point in DW
1.5	DW expansion factor
1500	GB online for DB
0.33	Unix comp factor
800	avg changes / sec [1]
28.5	avg rate of change - lo rate [2]
6048	total mnemonics [2]
477	hi rate parms [8]
150	10 sec averages [3]
300	30 sec averages [3]
5598	lo rate parms [2]
2,389	points/sec to flat files [1,8]
4,164	zero rate params [2]
0.386	average rate of change - zero rate [7]

Data Volume	GB/yr	1996*	2001	2006	2011
lo rate c-o in dw[7]	44	88	306	525	744
hi rate averages in dw [3]	38	76	265	454	644
full res hi rate to flat files [6,8]	405	811	2838	4865	6892
sum of c-o, avgs, and flat	487	974	3409	5844	8279
Number of Records	**M rec/yr**				
lo rate c-o	912	1,823	6,381	10,939	15,496
hi rate averages	789	1,578	5,523	9,467	13,412
sum	1,700	3,401	11,903	20,406	28,908

Bytes	Field
8	Start time
8	Stop time
2	Numeric ID (mnemonic)
4	Raw value
8	EU value
2	Flags

Notes
*All calculations assume H format, except estimation for first 5 yrs is 3:1 A format
These calculations do *not* account for random loss of signal. Data missed during an LOS is logged as a change for those parameters affected.
[1] approximation based on R Chu analysis of statistics on 2 major frames and from R Doxsey data
[2] derived from R Doxsey data
[3] Doxsey memo 10/26/95
[4] packed CDF blocks
[5] derived from R McMullan sample of AEDP subset
[6] assume hourly indexed flat files, CDF blocks, UNIX compressed
[7] assume "zero rate" changes at 3 hour average. These parameters must change sometime; longest data sample analyzed is 3 hrs.
[8] for this analysis assume "hi rate" applies to params greater than 1 Hz sample rate, which accounts for 477 parameters, with 143320 points per major frame (avg 2389 points per second).

Two candidate database schemes have been proposed; the first is indexed by mnemonic, and the second is indexed by time. Both schemes apply to changes-only data.

Pilot demonstrations of the mnemonic indexed scheme have resulted in indices on the order of 1.5 times greater volume than the data being stored. This is due mainly to the contradiction between the chosen index scheme and the fundamental assumptions underlying the design of the Red Brick product. Red Brick assumes that the data will be indexed on time and be predominately historical in nature. It is expected that some tuning of the database design can lower the index sizes.

A second concern has been identified in the mnemonic-indexed design. The tables are segmented by mnemonic, and are therefore not consistent in time. Additional processing overhead is required to update the tables.

An alternative indexing scheme has been proposed, one which attempts to maintain the temporal integrity of the database in order to simplify updates and segmentation. This approach also intends to minimize the number of tables in order to reduce the index overhead. This design approach is still in development.

This discussion has shown that the performance of the data warehouse product is closely coupled with the design of the database. Database design also impacts the format of the data to be stored, and calls into question the capability of the data warehouse product to perform this function. This issue underscores the necessity of designing the CCS archive in an open manner. The design of the data server and the choice of data in ancillary storage must be isolated from changes in database design or data warehouse product.

Ancillary Data Storage

Both options presented in the preceding section address reduction only of data which does not change often, making it highly reducible and a good candidate for the data warehouse. Dynamic data (that which changes often and is therefore sampled more frequently) must be extracted from the input by the preloader process, and stored as flat files in ancillary storage. The format of high-rate flat files will be FOF, with a file structure indexed by time. This data volume can be reduced using binary compression tools native to the operating system. Compression on the order of 3-to-1 can be expected on telemetry data formatted in FOF packets.

The action of splitting the input data into two subsets - low-rate changes-only and high-rate flat files - potentially introduces complexity and dependency into the data server. This creates subsets of data which cannot be tightly bound without introducing significant complexity into the data server and violating concepts of encapsulation.

In order to minimize complexity in the data server, relational operations and complex queries must be restricted to the data subset which is contained in the warehouse. This implies that any data which would benefit from a relational database operation must be reduced via some useful algorithm (e.g. averages, culled points, statistics). This raises a configuration control issue for the preloader, similar to that of the data warehouse.

Issue #3 - What is the control mechanism for defining parameters to be processed by the preloader. What is functionally required of the preloader if the warehouse content is modified (Issue #1) or if the database design or product changes. Will historical data be re-ingested if it is impacted by the change.

Intelligent queries might simplify the task of the data server in determining which data subset contains the data of interest. However, when a simple time-bounded, all-points extraction is requested for data which exists in the warehouse, the data server must still resolve the query and perform extra processing in order to reconstruct all-points from changes-only data.

Several concerns have been voiced regarding: possible drift of the time value during reconstruction; how to construct a query which will guarantee a hit in the warehouse; the complexity of the process which must assemble partial solutions from disparate sources; and whether telemetry playback is possible.

Operationally limiting queries addresses some of the complexity of managing multiple data sets. However, reconstruction of all-points from changes-only remains a concern. Given that the majority of the data will be maintained in flat files by default, it should be reasonable to evaluate the cost of storing *all* data in flat files. This cost can then be compared with the complexity and performance limitations of extracting time-domain data from a frequency-domain database.

Table 2 contains a comparison of data volumes for high-rate flat files versus all-points data in flat files. As indicated, the difference amounts to 141 GB per year, or approximately $5000 in optical storage media, or less than $700 in tape media, depending upon the choice of architecture. This projection assumes a typical 3-to-1 compression factor using UNIX native compress on FOF files. Current prices and storage densities are assumed.

Considering that the initial size and cost of the system is driven by the need to support high-rate data, the cost of storing an additional 140 GBytes per year is nominal. To make integration simpler and the system more sustainable, the complete archive can be implemented within an overall storage management system. This will be addressed in the following section.

This may appear to be storing 35% of the data redundantly, but it should be understood that the preloader functionally modifies the source data. There is no assurance that it will produce a reliable copy of the data for all cases. Data integrity becomes an issue for reconstruction, and the cost of addressing this issue can be significant.

Table 2 - Volume Comparison for Flat Files

161.445	avg data points / mframe [1]
20	mframes / sec
3228.9	data points / sec (pps)
8	bytes per packet header
16	bytes per data point
0.33	Unix comp factor
2388.67	hi rate points/sec to flat files [1,8]
20	GB per tape
$ 97	cost per tape
2.6	GB per optical
$ 63	cost per optical

Data Volume	GB/yr	1996*	2001	2006	2011
high-rate flat [6]	405	811	2838	4865	6892
all-points flat [6]	547	1094	3828	6562	9296
difference (all points - hi rate)	141	283	990	1,697	2,404

Costs	annual	1996	+5yr	+5yr	+5yr
additional disks required	54	109	272	272	272
differential cost of optical media	$ 3,427	$ 6,853	$ 17,133	$ 17,133	$ 17,133
additional tapes required [10]	7	14	49	99	184
differential cost of tape media	$ 686	$ 1,372	$ 4,801	$ 9,602	$ 17,833
cost of optical media, all points	$13,250	$26,499	$ 66,249	$ 66,249	$ 66,249
cost of tape media, all points	$ 2,652	$ 5,304	$ 18,564	$ 37,129	$ 68,953

Notes
*All calculations assume H format, except estimation for first 5 yrs is 3:1 A format
These calculations do *not* account for random loss of signal. Data missed during an LOS is logged as a change for those parameters affected.
[1] approximation based on R Chu analysis of statistics on 2 major frames and from R Doxsey data
[2] derived from R Doxsey data
[3] Doxsey memo 10/26/95
[4] packed CDF blocks
[5] derived from R McMullan sample of AEDP subset
[6] assume hourly indexed flat files, CDF blocks, UNIX compressed
[7] assume "zero rate" changes at 3 hour average. These parameters must change sometime; longest data sample analyzed is 3 hrs.
[8] for this analysis assume "hi rate" applies to params greater than 1 Hz sample rate, which accounts for 477 parameters, with 143320 points per major frame (avg 2389 points per second).
[10] annual cost of fresh tapes for rotation is included in projections thru 2011

Bytes	Field
8	Time
2	Numeric ID (mnemonic)
4	Raw value
8	EU value
2	Flags

Given the nominal cost of media to store the additional volume, it is recommended that all points of data be stored in flat files. This approach resolves several issues:

a) No reconstruction from changes-only is required for an all-points extraction. For applications that require all points, a simple time-bounded query will produce the solution set from flat files. For applications that expect changes only, changes can come directly from the warehouse, or can be reduced from all-points, as necessary.

b) No reconstruction of solution sets from disparate data sources is required. Operational limitations can be placed on queries and extraction requests such that no bridging of data sets by the data server is necessary.

c) Performance for time-domain analysis is maximized.

d) Performance for frequency-domain is maximized.

e) The system is more flexible in responding to a variety of user requests.

f) Accurate telemetry stream playback is possible, at minimal cost and complexity.

g) Partial or total reconstruction of the data warehouse is possible at any time.

h) The data server will initially retrieve telemetry from an all points, flat file format. The data warehouse will be a simple "enhancement" to this capability, rather than a complex redesign.

i) Changes to the database or warehouse are de-coupled from the data server processes which work with flat files. The risks to data server design and system operation are mitigated.

j) There is no question as to the integrity of the data. There is one complete archive which serves as the source of all engineering telemetry.

Frequency-domain analysis and complex queries will remain limited to the data warehouse. Time domain analysis can be supported by the warehouse, but only at a limited resolution; that which is supported by changes, averages, culled points, or statistics. It is expected that the warehouse will be used to relate events with conditions, or to analyze data at a low resolution. The results of this type of query will return a time frame which can be used to request a full-resolution extraction from the flat file archive, if necessary. Data extracted from flat files will always be limited to time-bounded, non-relational *data extraction* requests. For performance, reasonable limitations will be placed on the length of time requested from flat files.

Bridging the data sets will be the responsibility of the user, and may be simplified by tools at the user interface. Queries and data extraction requests can be combined only with some intelligent use of the user interface tools, the data server, and the archive.

It should be stressed that the data warehouse can be considered only a repository for data *products*. The entire data set exists, in native format, in ancillary flat files. The preloader is a tool which generates a *representation* of that data set to support efficient querying and sampling. This approach de-couples the data server and ancillary storage from the design of the data warehouse and preloader.

Archive Architecture

Conclusions reached in the preceding sections allow the archive system to be simplified and more easily partitioned for parallel development and implementation. Figure 1 presents a system data flow block diagram. The proposed system consists of three distinct elements: the data archive, the data warehouse, and the data server.

Data Archive Subsystem

The data archive consists of the storage media for the long-term data store, processes for staging and merging the input telemetry, and processes to manage the storage resource.

Functionally, the data archive receives FOF files from the FEP and stages them on local storage. Periodically, the merge function processes the data and stores it once on magnetic disk. At this point, the storage management function may begin its task of migrating data, performing backups, and distribution processing, as necessary. The data warehouse subsystem is notified that new data is available.

Physically, the data archive is comprised of the platform which hosts the merge and storage management processes. Data storage is made up of magnetic disks for staging and near term retrievals, and optical or tape media for long term archival.

Each element of the storage resource is intended to be modular and scaleable. Magnetic disk storage is sized to maintain a [five]-day staging area, a [100]-day repository of merged data, and overhead for system and user files. Near line media (tape or optical) is sized to maintain approximately 10 years of telemetry data, accessible by robotics. The remainder of the archive may be placed on the shelf, or additional jukebox capacity may be purchased beyond the year 2000.

Table 3 presents rough-order estimates for a candidate configuration. A packaged HSM system is recommended in order to minimize cost of integrating and maintaining the system long-term. Two options are provided for consideration: one based on optical storage, and the other based on magnetic tape (DLT). Both options assume that the existing magnetic disk storage system of 300 GB is available. The "magnetic disk" column is included as a placeholder, should this assumption change.

Table 3 - Cost Estimate for Data Archive Storage

	mag disk	tape	optical
HSM w/ software			
capacity thru 2001 (GB)	600	4,800	4,800
cost ($k)	(on-hand)	$234	$300
replacement media cost thru 2001 ($k)	n/a	$1	n/a
subtotal cost thru 2001 ($k)	-	$235	$300
capacity 2001 - 2011 (GB)	(no addl)	13,300	13,300
cost ($k)	-	$137	$313
replacement media cost 2001 - 2011 ($k)	n/a	$25	n/a
subtotal cost thru 2001 ($k)		$162	$313

Integrated Costs	**2001**	**2011**
Total cost, tape option ($k)	$235	$397
Total cost, optical option ($k)	$300	$613

Data Warehouse Subsystem

The data warehouse consists of the storage media dedicated to the database, the preloader process which extracts and builds the warehouse data subset, and the warehouse database product.

Functionally, the preloader process receives merged data files from the archive and extracts the data necessary to build the changes-only subset, averages, culled points, and statistics. Any modifications made to the data are contained only within the warehouse subsystem, and do not propagate back to the archive. The data warehouse product manages the storage system which is dedicated to the warehouse subsystem.

Physically, the data warehouse is comprised of the platform which hosts the data warehouse product. Warehouse storage is initially comprised of a magnetic disk farm, sized to contain all warehoused data online, and designed to be extended as volumes increase. This storage subsystem is completely independent of the archive subsystem, since the Red Brick product is presently incompatible with storage management systems. Red Brick also projects support for optical media in future releases, making the warehouse storage system open-ended and flexible.

The warehouse storage resource is intended to be modular and scaleable. Magnetic disk storage is sized to maintain all data online for the first several years. As the mission proceeds and volumes increase this disk farm may be expanded, or advances in storage technology may offer a cost savings to go to higher density devices, or data may migrate to optical. This decision need not be made until the year 2000 time frame. Table 4 presents rough-order estimates for a candidate configuration.

Table 4 - Cost Projection for Data Warehouse Storage

Magnetic RAID - SGI	thru 2001	thru 2011
GB per drive	4.3	4.3
GB per array	17.2	17.2
GB total storage	600	1500
daily data rate (GB)	0.223	0.223
days storage online	2685	6712
RAID 5 factor	0.8	0.8
gross core encl cost	$ 206,800	$ 517,000
gross exp unit cost	$ 12,576	$ 31,440
gross addl array cost	$ 464,475	$ 1,168,225
gross addl cntlr cost	$ 47,884	$ 47,884
total estimated cost	$ 731,735	$ 1,764,549
cost/MB	$ 1.22	$ 1.18

Data Server Subsystem

The data server consists of the processes which interface with the long-term archive, the data warehouse, and the end user, as well as the database which identifies the status and location of various data sets.

Functionally, the data server submits appropriately-formatted requests to the archive and the warehouse as required and formats the received data for transmission to the user.

Physically, the data server will likely reside on the platform which hosts the archive subsystem.

Summary and Recommendation

It is the intent of this study to identify an approach to formatting and storing the archive data. The solution must be compatible with the data warehouse product, and must provide a reasonable balance among performance, system resources, system complexity.

The two popular options for data storage, changes-only and all-points, have opposing advantages and disadvantages, making neither one a clear choice. Each one represents an advantage to its own class of users. Performance of the system can be tuned to one type of query only at a cost to the other. Several variations of these options were considered in this analysis, and it became apparent to the community that the final format needs to be some combination of the two options.

The data warehouse product represents a valuable tool in accessing and analyzing a significant amount of data. Its ability to relate events and data represents a significant enhancement to the capabilities of the current system. There is no question over the utility of the data warehouse as an analysis tool.

However, no database product can effectively *manage* all engineering data as it is received. It is essential that the amount of data to be maintained in the database be reduced to its minimum. This implies that some judgment be made regarding various aspects of the data, such as its importance, relevance, utility, or value. Manipulating the content of the data in this way is unavoidably arbitrary, and it necessitates control processes and mechanisms to ensure fairness to the users of the system. Furthermore, a majority of the data, over 65%, cannot be effectively reduced. This approach still requires a large data archive - on the order of 400 Gbytes per year, in a compressed flat file format.

On the other hand, all points of data has its value. Some users require a complete data set for plotting, statistical analysis, and telemetry playback. Modification of data for the sake of volume reduction imparts a penalty on reconstruction of that data for these uses. There is also concern over the accuracy of the reconstruction process, and the integrity of the data it represents. These lessons were learned as the current PRS system matured. All points of data, in compressed flat file format, amounts to 550 Gbytes per year.

The recommended solution is to provide a long-term archive of all points of data in a flat file format, which can be compressed to a reasonable volume without impacting the tools which access the data. This data can then be used as input to the data warehouse, continuously or on demand, whereby only data which is of value to those who use the warehouse tool is extracted. The annual differential cost of providing an all points flat file archive, versus a high-rate flat file subset, ranges between $800 and $5000, depending upon storage medium. *The cost of the system is virtually the same for changes-only and all-points*.

This approach resolves several issues:

a) No reconstruction from changes-only is required for an all-points extraction. For applications that require all points, a simple time-bounded query will produce the solution set from flat files. For applications that expect changes only, changes can come directly from the warehouse, or can be reduced from all-points, as necessary.

b) No reconstruction of solution sets from disparate data sources is required. Operational limitations can be placed on queries and extraction requests, or user interface tools can be developed, such that no bridging of data sets by the data server is necessary.

c) Performance for time-domain analysis is maximized.

d) Performance for frequency-domain is maximized.

e) The system is more flexible in responding to a variety of user requests.

f) Accurate telemetry stream playback is possible, at minimal cost and complexity.

g) Partial or total reconstruction of the data warehouse is possible at any time.

h) The data server will initially retrieve telemetry from an all points, flat file format. The data warehouse will be a simple "enhancement" to this capability, rather than a complex redesign.

i) Changes to the database or warehouse are de-coupled from the data server processes which work with flat files. The risks to data server design and system operation are mitigated.

j) There is no question as to the integrity of the data. There is one complete archive which serves as the source of all engineering telemetry.

Figure 1 shows a block diagram of the system. The recommended system consists of three elements: the data archive, the data warehouse, and the data server. This partitioning supports distinct interfaces and simplifies development. Data flowing among the archive, warehouse, and data server can be easily managed at the interfaces and controlled via internal ICDs. Modifications made to one subsystem will not impact the design of the others. This partitioning also supports replication of essential elements for other facilities, maximizing the benefits of reuse, and minimizing the costs to the remote facility.

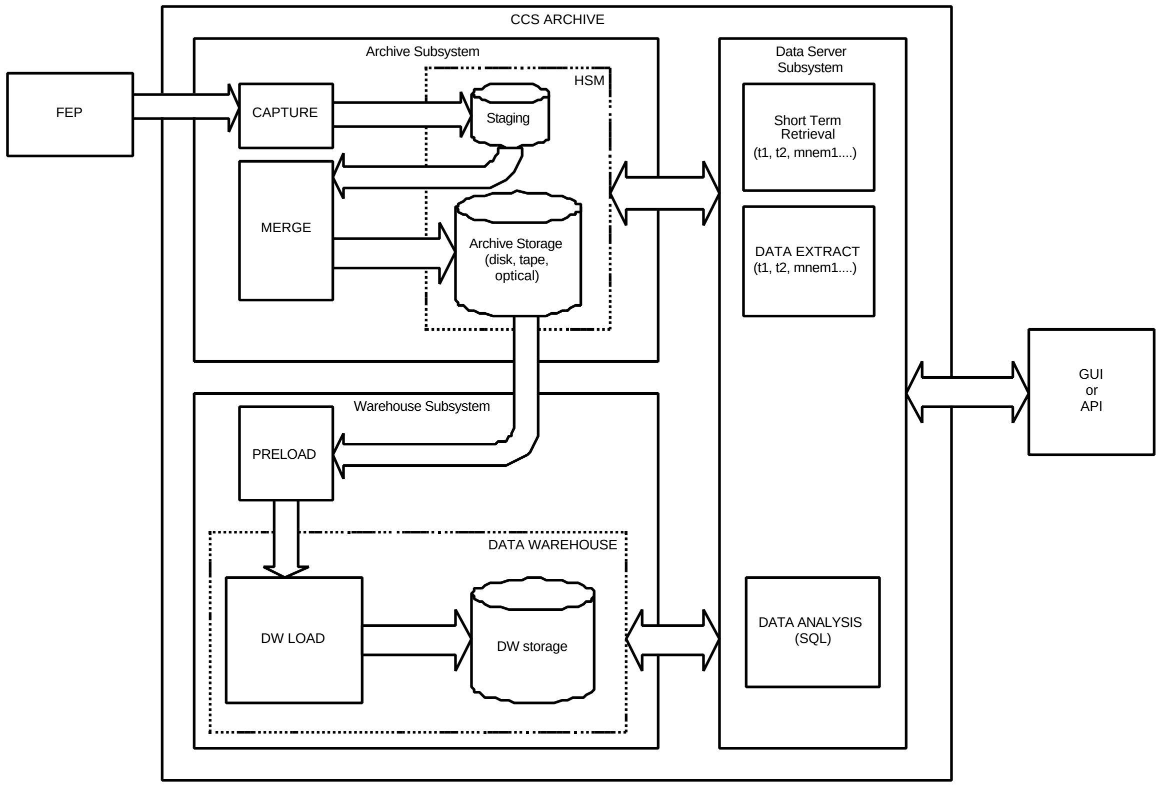

Figure 1 - Telemetry Archive Interfaces

Table 5 summarizes the total estimated cost for the system recommended above. This estimate compares optical and tape options. Costs are rough-order, non-competitive, for purposes of comparison only. For each option, it is assumed that the current 600 GB RAID array will be made available to the archive system, and that the data warehouse system will be comprised of similar hardware in the capacities shown. Both estimates provide sufficient capacity for all data to be kept accessible to robotics. Costs will be lower if archive is not expanded, and aged data is placed on the shelf. Cost savings in hardware would be offset by operations costs to support feeding disks and/or tapes. All costs assume current pricing and capacities - industry trends currently double storage capacity every 4 years.

Table 5 - Summary Cost Comparison

Summary Cost	**thru 2001 capacity (GB)**	**cost ($k)**	**2001-2011 addl cap (GB)**	**addl cost ($k)**
Option 1 - DLT Archive				
DW RAID	600	$731	900	$1,096
Archive RAID	600	(on hand)	-	-
Archive DLT [1]	4,800	$235	8,500	$162
TOTAL	6,000	$966	9,400	$1,258
Option 2 - Optical Archive				
DW RAID	600	$731	900	$1,096
Archive RAID	600	(on hand)	-	-
Archive Optical [1]	4,800	$300	8,500	$313
TOTAL	6,000	$1,031	9,400	$1,409

[1] incl. HSM software and integration costs

Figure 2 shows how the conceptual design fits into the overall CCS data flow.

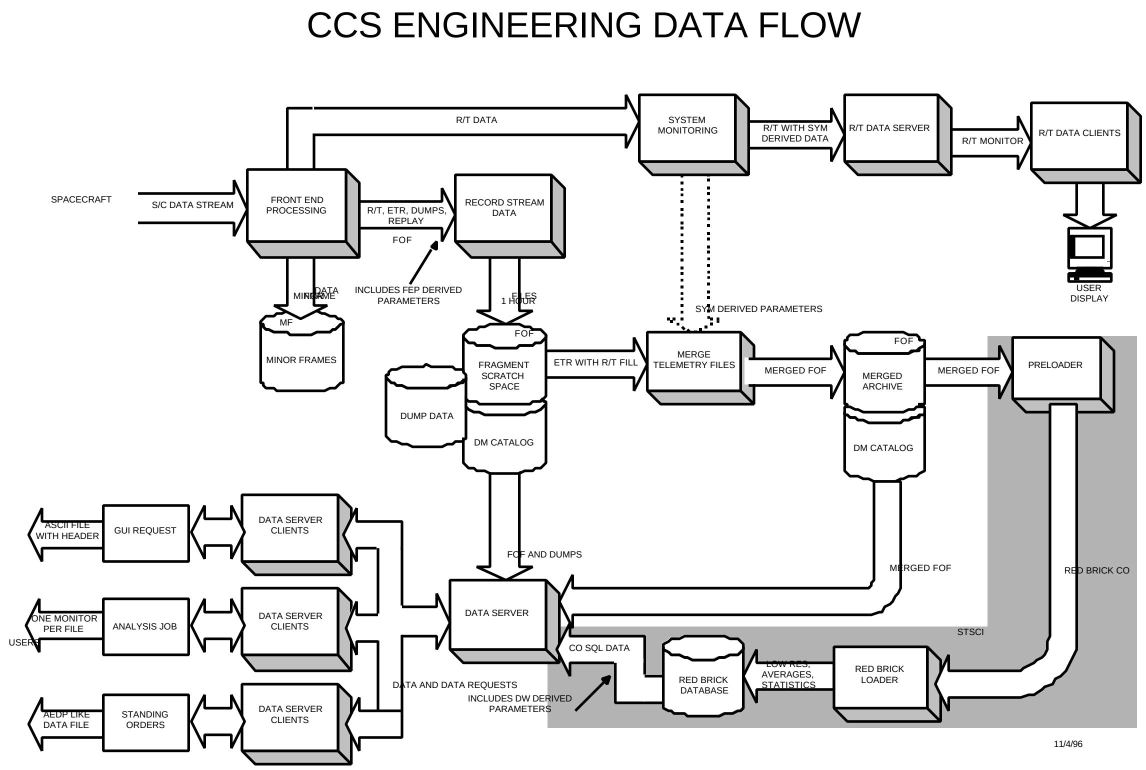

Figure 2 - CCS Data Flow Block Diagram

Three issues have been identified:

Issue #1 - Configuration management of the content of the database must be resolved. How are parameters selected for inclusion or exclusion. What criteria are applied to the choice. How will exceptions be resolved and implemented.

Issue #2 - The database design is currently in development. Preliminary results have shown limitations. Choice of data format and selection of parameters is dependent upon database design, and the design and capabilities of the data warehouse may change at any time. This represents a risk to the design of the data server. Designs of the data server and data warehouse must be encapsulated in order to avoid impacting other elements of the archive subsystem.

Issue #3 - What is the control mechanism for defining parameters to be processed by the preloader. What is functionally required of the preloader if the warehouse content is modified (Issue #1) or if the database design or product changes. Will historical data be re-ingested if it is impacted by the change.

These issues can be resolved as the design of the system progresses and is implemented.

Appendix A

The Others

Several alternative approaches to data reduction were considered. The following options were excluded from detailed evaluation for the reasons specified.

All points in data warehouse

The Red Brick product is designed to be able to address up to 2^{42} rows of data. However, the data which is manageable at any one time is functionally limited to about 1.5 billion rows (this limitation has not been market tested). Beyond this, the indices grow too large, and performance degrades.

This ceiling limits active access to about 5 days worth of all-points data. While all data can theoretically be kept in inactive segments, it is impractical to do so.

All changes

Changes only, in its simplest form, involves reducing the data by extracting all data which is repeated in consecutive packets. In general, this option reduces storage volume and cost, but increases complexity of the data server code. It does not necessarily affect the efficiency of the overall storage and retrieval process - pre-loading and data retrieval become more efficient, but multiple queries and reconstruction processing in the data server offset those savings. Queries over large time intervals become more manageable, but "data mining" capability (queries on associated data or more focused intervals) suffers in performance.

The data warehouse record limitation discussed under all points applies also to this option. All changes would fill the 1.5 billion row “active” table in 21 days. This remains impractical for the data warehouse.

Modified FOF

There are a number of techniques which might be applied to reduce the size of the data fields in the FOF. One example utilizes a scheme which codes the spacecraft time into a five byte field.

A second approach assumes that the Raw and EU fields of the FOF are only meaningful for those parameters to which calibration coefficients are applied - analogs. These two fields are redundant for discrete, bi-level, and memory parameters (or one field is null).

By utilizing a variable field definition, the volume can be reduced by the amount of redundant or meaningless data. This comes at the cost of complexity to the data storage and extraction software. This option may still be worth investigating for reduction of the archive volume. It has no impact on the data warehouse.

Packed bilevels

This option is based on the observation that for each bit in a bilevel, the data warehouse must store 20 bytes. A method which packs 8 bilevels per byte in the DW would reduce the all points estimate by as much as 1500 mnemonics. However, this assumes an even distribution of bilevels across all minor frames (it is essential that all bilevels packed under one entry be associated in the same half-minor-frame). It is not known what the bilevel distribution truly is, nor is it known yet how many points per second this reduces the storage volume. Format changes also result in a different distribution, which must be accounted for. This option introduces low to moderate complexity in the preloader phase in order to pack the bilevels. The data server design would see moderate to high complexity since queries would be more complex, and expansion and reconstruction would be required. This option is considered less practical than other approaches, and will not be evaluated fully.

Binary compression for data warehouse

Won't work with the data warehouse since RB does not work on compressed data and is not presently designed to make OS calls to compress and decompress. The only way compression would be possible is if: a) it is done at the file level by the operating system (or HSM), and decompressed when copied to online medium; or b) it is done by the storage hardware on the fly. Any compression performed will affect the retrieval performance. Benchmarks must be run to determine whether the I/O performance will meet the RB response time requirements.

Red Brick compression

A misnomer, really. Red Brick converts from ASCII to binary, providing a 10% reduction. The estimated 30% DW expansion factor includes this reduction.